Abstract
The amount of software in many systems increases exponentially. This increase impacts the reliability of these systems. In the source code of software many hidden faults are present. These hidden faults can transform into errors during the system life cycle, due to changes in the system itself or in the context of the system.

We will discuss the current trends and potential directions for future solutions of an increasing reliability problem.
Mission
To advance industrial innovation and academic excellence in embedded systems engineering (ESE).

Vision
ESI and its partners create and apply world-class ESE methods.

7 Founders:
Industry (Philips, ASML, Océ)
Universities (Twente, Delft, Eindhoven)
Knowledge Institutes (TNO)
Embedded Systems Engineering

- performance
- reliability
- evolvability

in relation to all other system qualities: security, power, cost, size, et cetera
Industry as Laboratory

source of inspiration
application playground
industry

challenging problems
apply new engineering methods
research

improve
evaluate

observe results
hypothesis
Who is Gerrit Muller?

- CT
- OSS
- NM
- X-ray

- MR

- EasyVision

- Gaudí

Philips Medical

- 1980
- 1990
- 2000

ASML

- Philips NatLab
- ESI

Buskerud

Industrial experience
- time pressure
- pragmatics
- sales
- cost constraints
- products
- lots of people

Research
- reflection
- evidence
- exposure
- education

Reliability of Software Intensive Systems
version: 0.2
March 6, 2013
FIESAwhoisGerrit
www.gaudisite.nl

deze lezing:

Exploration of Reliability

Introduction
  + ESI
  + Speaker

Exploration
  + trends
  + consequences for reliability
  + potential solutions

Research Projects
  + Trader
  + Tangram
  + Ideals
  + Boderc

Conclusion
Trends in Embedded Systems

How to survive in innovative domains?

- Fast moving market
- Increased integration
- Complex value chains
- Fast moving technology

Reliability of Software Intensive Systems
Gerrit Muller
version: 0.2
March 6, 2013
ECMAproblem
Increased Team Size

Reliability of Software Intensive Systems
10  Gerrit Muller

version: 0.2
March 6, 2013
DYOTeamsise
Number of Faults Proportional With Code Size

Based on average 3 errors/kloc

Reliability of Software Intensive Systems
Pilot announces a flight delay, due to computer problems. A complete reset is required. The flight entertainment system also show a reset: a complete Linux boot. This reboot hangs: server xxx not found.
How to Make SW Intensive Systems Reliable

within cost&time
improve reliability
without performance penalty
with increasing requirements
increasing #suppliers #sites, et cetera

reduce functionality
more/longer testing
more procedures
reduce code
improve way-of-working
more standardization
improve recovery
improve detection
improve robustness

more formality
more automation
early integration
shared understanding
more feedback
tolerant interfaces
robust patterns
Reliability Research

Introduction
  + ESI
  + Speaker

Exploration
  + trends
  + consequences for reliability
  + potential solutions

Research Projects
  + Trader
  + Tangram
  + Ideals
  + Boderc

conclusion
Example Trader Project

**Overview**

**Trader** Television Related Architecture and Design to Enhance Reliability

**Objective:** Develop method & tools to ensure reliable consumer electronics products

**Research agenda:** System Reliability

**Domain:** Digital TV

**CIP:** Philips Semiconductors

**Partners:** Philips CE, Tass, and Research DTI, IMEC, TUD, TU/e, UL, UT, ESI

**Timeline:** 9/2004 – 9/2008, 22 Fte

---

**Industrial Relevance**

Poor reliability has severe business impact

- Customer expectation of TV reliability is high
- Little tolerance for technical problems
- 100% fault-free design is not achievable
- High volume market implies high risks if reliability problems occur
  - Low product margin leaves no buffer for service costs
  - Service center costs multiplied by number of complaints
  - Market share reduction likely, i.e. customers buy another brand
- (On) Time to market is critical
  - Missing fixed shipping gates costs millions of dollars

---

**Trader Domain Trends**

TV complexity increase follows the PC world

from “Display movies over antenna” to “Display anything over everything”

- **Broadcast**
  - ATSC, DVB, ISDB, Analog
  - Terrestrial, Satellite, Cable

- **Connectivity**
  - Cameras, JPEG, flash cards, HDD, MP3, Web browser, Ethernet, USB, etc...

---

**User Perceived Reliability**

- **Objective**
  - Determine the user perceived severity of a product failure mode

- **Methodology**
  - Create a model considering relevant factors
    - User-perceived loss of functionality
    - User-perceived reproducibility
    - Failure frequency
    - Work-around difficulty
    - User-group characteristics
    - Failure characteristics
  - ... Validating model
  - Evaluate and suggest system failure recovery strategies
    - The recovery strategy may not annoy the user even more!

---

**Table: Aspects**

<table>
<thead>
<tr>
<th>Aspects (all depends on user)</th>
<th>1</th>
<th>2</th>
<th>3 (JPEG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function importance</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Reproducibility</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Solvability</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Loss of Function / time / Behavior</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Total:</td>
<td>576</td>
<td>199</td>
<td>9</td>
</tr>
</tbody>
</table>

---

Reliability of Software Intensive Systems

version: 0.2

March 6, 2013

Gerrit Muller
Increasing Code Size in Televisions

Moore's law

From: COPA tutorial, Rob van Ommering

Reliability of Software Intensive Systems

version: 0.2
March 6, 2013
LWAmooresLawRvO
Research: System Awareness to Improve Reliability

user input → system → system output

- awareness
- monitor
- correction
- customer expectations
- system failure model

Reliability of Software Intensive Systems

version: 0.2
March 6, 2013
RSIStraderAwareness
Expose product weaknesses at design time

- **Source code analysis**
  - Identify hotspots in code
  - Consider impact to user-visible behavior: *prioritize warnings*

- **Software architecture reliability analysis**
  - Techniques to identify failure-prone components
  - Evaluation of architectural alternatives and trade-offs
Quality Degradation Caused by Shit Propagation

needed code

needed code

copy
paste
modify

bad code

new needed code

code not relevant for new function

repair code

bad code

new bad code
Example of Shit Propagation

Class Old:
    capacity = startCapacity
    values = int(capacity)
    size = 0

    def insert(val):
        values[size]=val
        size+=1
        if size>capacity:
            capacity*=2
            relocate(values, capacity)

Class New:
    capacity = 1
    values = int(capacity)
    size = 0

    def insert(val):
        values[size]=val
        size+=1
        capacity+=1
        relocate(values, capacity)

Class DoubleNew:
    capacity = 1
    values = int(capacity)
    size = 0

    def insert(val):
        values[size]=val
        size+=1
        capacity+=1
        relocate(values, capacity)

        def insertBlock(v,len):
            for i=1 to len:
                insert(v[i])

Reliability of Software Intensive Systems

version: 0.2
March 6, 2013
BLOATshitPropagationExample
Example Tangram Project

Integration & Test

Early Integration

- Incremental
- Model Based
- Model ↔ Realization
- Infrastructure
- NDDS, Matlab, Chi

Results:
- Early Model Based Integration and Testing in AGILE
- Problem found in Sn CoMo before review
Moore’s Law

Reliability of Software Intensive Systems

Gerrit Muller

version: 0.2
March 6, 2013
ASMLmooreLaw
Challenge: Exponential Increase

- Performance
- Feedback and adjustments
- Imaging overlay
- Productivity
- HW and SW components
- Innovation
- Complexity
- Reliability
- Robustness

Reliability of Software Intensive Systems

March 6, 2013
ASML problem IRC
4 Views on a Waferstepper

**subsystems**

- Laser light source
- Illuminator beam shaping
- Reticle stage positioning
- Reticle handler input/output
- Lens projection
- C&T contamination, temperature
- Wafer stage positioning
- Wafer handler input/output

**control hierarchy**

- Laser
- Illuminator
- Lens
- Measurement
- C&T
- Reticle stage
- Reticle handler
- Wafer stage
- Wafer handler

**kinematic**

- Vertical motion
- Horizontal motion
- Dynamic exposure through slit

**physics/optics**

- Laser
- Illuminator
- Sensor
- Pulse-freq, bw, wavelength, ...
- Uniformity
- NA aberrations transmission

Reliability of Software Intensive Systems

version: 0.2
March 6, 2013
EBMIsystemDiagrams
Dynamic simulation of integration & test approach:
1. Create model (modules, interfaces, faults, tests)
2. Execute model at any time
3. Balancing based on time, cost and remaining risk.

Balancing functionality, quality and time/cost
(over 20% reduction of integration & test time)
Example Ideals Project

code size \[\xrightarrow{\text{reduce by}}\] cross cutting concerns

Results: Parameter Check

- Automatic replacement of current parameter checking and tracing idiom with specific aspect oriented idiom.

- Code size reduction of 80% for parameter checking idiom (7% reduction of total module)

- Improved locality
Example Boderc Project

31x5E

2050

2090
Boderc Goal

Boderc goal = 
A specific methodology to predict system performance and analyze, discuss, document, and communicate throughput, quality and power computing response time based on modeling through cost, project duration, process, and people within industrial constraints and restricted design space.
Shared Understanding by Modeling

- $10^0$
- $10^1$
- $10^2$
- $10^3$
- $10^4$
- $10^5$
- $10^6$
- $10^7$

**Number of details**

- System
- Multi-disciplinary
- Mono-disciplinary

**Factors**

- Back-of-the-envelop
- Formula based
- Executable
Many Models Needed to Understand System

6. Kinematic modeling

7. Thermo modeling

8. Control architecture

9. Virtual printer models

10. Stepper motors

small, simple, goal-driven models

shorter cycle time, less cycles

shorter product creation lead time
Coverage of Reliability by ESI Projects

- within cost&time
- improve reliability
  - without performance penalty
  - with increasing requirements
  - increasing #suppliers #sites, et cetera

- reduce functionality
  - improve way-of-working
    - reduce code
      - more standardization
      - improve recovery
      - improve detection
      - improve robustness
    - more formality
      - more automation
    - early integration
      - shared understanding
      - more feedback
      - tolerant interfaces
      - robust patterns

- more/longer testing
- more procedures
- improve way-of-working
- more formality
- more automation
- early integration
- shared understanding
- more feedback
- tolerant interfaces
- robust patterns

Boderc
Tangram
Ideals
Tangram
Tangram
Trader
Trader
Trader
Towards a Conclusion, Some more Trends

Introduction
  + ESI
  + Speaker

Exploration
  + trends
  + consequences for reliability
  + potential solutions

Research Projects
  + Trader
  + Tangram
  + Ideals
  + Boderc

Conclusion
Applications depend on chain of systems

- Users
- Infotainment appliance
  - Watch video
  - Browse photo's
  - Calendar
  - And much more...

Diagram:
- Users
- Home Server
- Network Providers
- Service Providers
- Content Providers
Interoperability: systems get connected at all levels
Multi dimensional interoperability

integrating multiple applications
clinical analysis
clinical support
administrative
financial
workflow

in multiple languages
cultures
USA, UK,
China, India,
Japan, Korea
France, Germany
Italy, Mexico

delivered by multiple vendors
Philips
GE
Siemens

based on multiple media, networks
DVD+RW
memory stick
memory cards
bluetooth
11a/b/g
UTMS

and multiple standards
Dicom
HL7
XML

and multiple releases
R5
R6.2
R7.1
Interopearbility Trends and Research Challenges

<table>
<thead>
<tr>
<th>Trends</th>
<th>(Partial) Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>market dynamics</td>
<td>globalization</td>
</tr>
<tr>
<td></td>
<td>hype waves</td>
</tr>
<tr>
<td></td>
<td>Moore's law</td>
</tr>
<tr>
<td>interoperability</td>
<td>emerging behavior</td>
</tr>
<tr>
<td></td>
<td>future vs legacy</td>
</tr>
<tr>
<td></td>
<td>heterogeneous vendors</td>
</tr>
<tr>
<td></td>
<td>dynamics (continually changing)</td>
</tr>
<tr>
<td>reliability</td>
<td>complexity</td>
</tr>
<tr>
<td></td>
<td>heterogeneity</td>
</tr>
<tr>
<td></td>
<td>dynamics</td>
</tr>
<tr>
<td></td>
<td>#engineers involved</td>
</tr>
<tr>
<td></td>
<td>standards</td>
</tr>
<tr>
<td></td>
<td>economical interests</td>
</tr>
<tr>
<td></td>
<td>latency due to slow process</td>
</tr>
<tr>
<td></td>
<td>most fundamental solution</td>
</tr>
<tr>
<td></td>
<td>design patterns</td>
</tr>
<tr>
<td></td>
<td>in system feedback</td>
</tr>
<tr>
<td></td>
<td>human-in-the-loop feedback</td>
</tr>
<tr>
<td></td>
<td>semantic understanding</td>
</tr>
<tr>
<td></td>
<td>containment</td>
</tr>
<tr>
<td></td>
<td>gateway</td>
</tr>
<tr>
<td></td>
<td>new research challenges!</td>
</tr>
</tbody>
</table>

new research challenges!
Conclusion

Based on average 3 errors/kloc

- less code
- reduce impact of hidden faults on system reliability

- less errors/kloc

man-years and LOC
(lines of code) per product


Reliability of Software Intensive Systems

version: 0.2
March 6, 2013